Anytime Approximate Bi-Objective Search

Anonymous

Abstract

The Pareto-optimal frontier for a bi-objective search problem instance consists of all solutions that are not worse than any other solution in both objectives. The size of the Pareto-optimal frontier can be exponential in the size of the input graph, and hence finding it can be hard. Some existing works leverage a user-specified approximation factor ε to compute an approximate Pareto-optimal frontier that could be significantly smaller than the Pareto-optimal frontier. In this paper, we propose an anytime approximate bi-objective search algorithm, called Anytime Bi-Objective A*-ε (A-BOA$^*_\varepsilon$). A-BOA$^*_\varepsilon$ is useful when deliberation time is limited. It first finds an approximate Pareto-optimal frontier quickly, iteratively improves it while time allows, and eventually finds the Pareto-optimal frontier. It efficiently reuses search efforts from previous iterations and makes use of a novel pruning technique. Our experimental results show that A-BOA$^*_\varepsilon$ significantly outperforms baseline algorithms that do not reuse previous search efforts, both in terms of runtime and number of node expansions. In fact, the most advanced variant of A-BOA$^*_\varepsilon$ even slightly outperforms BOA*, a state-of-the-art bi-objective search algorithm, for finding the Pareto-optimal frontier. Moreover, in limited time, A-BOA$^*_\varepsilon$ finds solution sets that collectively approximate the Pareto frontier much better than the solutions found by BOA*.

1 Introduction and Related Work

Bi-objective graph search is a generalization of single-objective graph search used for shortest-path computations. We are given a directed graph with two costs annotating each edge, a specified start vertex, and a specified goal vertex. Given a solution path π, we use $c_1(\pi)$ and $c_2(\pi)$ to denote the accumulated first and second costs on the edges of π, respectively. A solution path π is considered to be better than, i.e., to *dominate*, another solution path π' if and only if (i) $c_1(\pi) \leq c_1(\pi')$ and $c_2(\pi) < c_2(\pi')$ or (ii) $c_1(\pi) < c_1(\pi')$ and $c_2(\pi) \leq c_2(\pi')$. In bi-objective search, a typical task is to find the Pareto-optimal frontier, which consists of all those solution paths that are not dominated by any other solution path.

Bi-objective graph search is important in many domains such as transportation and robotics (Bronfman et al. 2015; Bachmann et al. 2018; Fu et al. 2019; Fu, Salzman, and Alterovitz 2021). In such domains, problems often have two objectives that cannot be optimized at the same time. For example, in the Hazardous Material Transportation problem (Bronfman et al. 2015), we are required to plan a route for transporting hazardous material while considering the travel distance as well as the risk of exposure for residents.

State-of-the-art algorithms such as BOA* (Hernandez et al. 2020), NAMOA* (Mandow and De La Cruz 2010), and NAMOA*dr (Pulido, Mandow, and Pérez-de-la Cruz 2015) can be used to compute the Pareto-optimal frontier. However, as the number of solutions in the Pareto-optimal frontier can be exponential in the size of the input graph (Ehrgott 2005; Breugem, Dollevoet, and van den Heuvel 2017), the problem of finding the Pareto-optimal frontier is intrinsically hard. Therefore, exact approaches often take long running times. To this end, an alternative approach calls for computing an approximation of the Pareto-optimal frontier (Warburton 1987; Breugem, Dollevoet, and van den Heuvel 2017; Tsaggouris and Zaroliagis 2009). Here, we are given a user-specified approximation factor $\varepsilon \geq 0$. A path π *ε-dominates* another path π' if each cost component of π is less than or equal to $(1 + \varepsilon)$ times the respective cost component of π'. An ε-approximate Pareto-optimal frontier is a set of solution paths Π_ε such that any Pareto-optimal solution path is ε-dominated by some path in Π_ε. In structured real-world problems, the ε-approximate Pareto-optimal frontier can have significantly fewer solution paths than the Pareto-optimal frontier, for even a small ε. For instance, in road networks, existing work (Goldin and Salzman 2021) has already demonstrated that search algorithms can find an ε-approximate Pareto-optimal frontier much faster than the Pareto-optimal frontier.

Another dimension to the characterization of a search algorithm is its *anytime* behavior. In anytime search, we are interested in quickly finding a "good" solution and progressively finding better solutions as time allows (Likhachev, Gordon, and Thrun 2003; van den Berg et al. 2011; Stern et al. 2014; Cohen et al. 2018). Ideally, an anytime search algorithm exhibits the "diminishing returns" property with regard to the quality of the current solution against increasing time, eventually converging to an optimal solution. Such an algorithm is useful when deliberation time is limited or unknown when a query is provided.

In this paper, we present an anytime approximate bi-objective search algorithm, called Anytime Bi-Objective A*-ε (A-BOA$^*_\varepsilon$). A-BOA$^*_\varepsilon$ has the characteristics of an approximate bi-objective search algorithm as well as an anytime search algorithm. It derives its anytime behavior from progressively tightening the approximation factor.[1] A-BOA$^*_\varepsilon$ starts by quickly finding an initial approximate Pareto-optimal frontier, subsequently finds more solution paths to improve the approximation factor, and eventually finds the entire Pareto-optimal frontier.

Any approximate bi-objective search algorithm can be naively converted into an anytime variant by invoking the search algorithm for progressively smaller values of ε. However, this methodology is inefficient for the reason that search efforts are redone across different values of ε. A-BOA$^*_\varepsilon$ revives this methodology by reusing previous search efforts. It does sufficient bookkeeping to allow each invocation of the search algorithm to resume the search efforts of the previous invocation. In doing so, it is significantly more efficient than the naive attempt. A-BOA$^*_\varepsilon$ also employs a novel pruning technique to further improve its efficiency: This pruning technique is based on a heuristic function designed to estimate a weighted sum of the two costs.

In later sections, we first provide proof sketches for the correctness and convergence properties of A-BOA$^*_\varepsilon$. We then compare it empirically against BOA* and two baseline algorithms that are derived from existing approximate bi-objective search algorithms without reusing previous search efforts. Our experimental results show that A-BOA$^*_\varepsilon$ significantly outperforms the baseline algorithms, both in terms of runtime and number of node expansions. In fact, the most advanced variant of A-BOA$^*_\varepsilon$ even slightly outperforms BOA* for finding the Pareto-optimal frontier. Moreover, in limited time, A-BOA$^*_\varepsilon$ finds solution sets that collectively approximate the Pareto frontier much better than the solutions found by BOA*.

2 Terminology

In this paper, we use boldface font to denote 2-tuples (i.e., pairs). We use $v_i, i \in \{1, 2\}$ to denote the i-th component of tuple $\mathbf{v}$. The addition of two 2-tuples $\mathbf{v}$ and $\mathbf{v}'$ is defined as $\mathbf{v} + \mathbf{v}' = (v_1 + v'_1, v_2 + v'_2)$. We define the following types of domination:

1. $\mathbf{v} \preceq \mathbf{v}'$ denotes that $v_1 \leq v'_1 \wedge v_2 \leq v'_2$. In this case, we say that $\mathbf{v}$ *weakly dominates* $\mathbf{v}'$.
2. $\mathbf{v} \prec \mathbf{v}'$ denotes that $\mathbf{v} \preceq \mathbf{v}' \wedge \mathbf{v} \neq \mathbf{v}'$. In this case, we say that $\mathbf{v}$ *(strictly) dominates* $\mathbf{v}'$.
3. An *approximation factor* ε is a non-negative real number. $\mathbf{v} \preceq_\epsilon \mathbf{v}'$ denotes that $v_1 \leq (1+\varepsilon)v'_1 \wedge v_2 \leq (1+\varepsilon)v'_2$. In this case, we say that $\mathbf{v}$ *ε-dominates* $\mathbf{v}'$. Note that, when $\epsilon = 0$, ϵ-domination is equal to weak domination.

A *(bi-objective) search graph* is a tuple $\langle S, E, \mathbf{c}\rangle$, where S is a finite set of *states* and $E \subseteq S \times S$ is a finite set of *edges*. Cost function $\mathbf{c} : E \to \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ maps an edge to two non-negative real numbers. We define $c_i : E \to \mathbb{R}_{\geq 0}, i \in \{1, 2\}$, as the function that maps an edge e to the i-th component of $\mathbf{c}(e)$. A *path* is a sequence of states $\pi = [s_1, s_2 \dots s_\ell]$ such that $\langle s_j, s_{j+1}\rangle \in E$ for all $j \in \{1, 2 \dots \ell - 1\}$. The cost of path π is $\mathbf{c}(\pi) = \sum_{j=1}^{\ell-1} \mathbf{c}(\langle s_j, s_{j+1}\rangle)$. We use $s(\pi)$ to denote the *ending state* of π, i.e., s_ℓ. We use $succ(s) = \{s' \in S | \langle s, s'\rangle \in E\}$ to denote the successors of state s. By *extending* π with an edge $\langle s_\ell, s_{\ell+1}\rangle$, we can obtain a new path $[s_1, s_2 \dots s_\ell, s_{\ell+1}]$. We say a path π' *extends* another path π if and only if π' can be obtained by applying a sequence of extend operations on π.

A *(bi-objective search) problem instance* is a tuple $P = \langle S, E, \mathbf{c}, s_{\text{start}}, s_{\text{goal}}\rangle$, where $\langle S, E, \mathbf{c}\rangle$ is a search graph, $s_{\text{start}} \in S$ is the *start state*, and $s_{\text{goal}} \in S$ is the *goal state*. A path is a *solution* for problem instance P if and only if it is a path from s_{start} to s_{goal}. For problem instance P, a *heuristic function* $\mathbf{h} : S \to \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ is an estimation of the cost of a path from a given state to s_{goal}. $\mathbf{h}$ is consistent if and only if $\mathbf{h}(s_{\text{goal}}) = \mathbf{0}$ and $\mathbf{h}(s) \preceq \mathbf{c}(\langle s, s'\rangle) + \mathbf{h}(s')$. In this paper, we limit the discussion to consistent heuristic functions only. For any path π from s_{start} to some ending state $s(\pi)$, its $\mathbf{g}$-value is defined as $\mathbf{c}(\pi)$, and its $\mathbf{f}$-value is defined as $\mathbf{f}(\pi) = \mathbf{g}(\pi) + \mathbf{h}(s(\pi))$.

Let π and π' be two paths beginning with state s_{start}. We say that π *dominates (resp. weakly dominates and ε-dominates)* π' if and only if $\mathbf{f}(\pi) \prec \mathbf{f}(\pi')$ (resp. $\mathbf{f}(\pi) \preceq \mathbf{f}(\pi')$ and $\mathbf{f}(\pi) \preceq_\epsilon \mathbf{f}(\pi')$). For a problem instance P, a *Pareto-optimal solution* is a solution that is not dominated by any other solution of P. The *Pareto-optimal frontier* Π^* is the set of all Pareto-optimal solutions, and a *cost-unique Pareto-optimal frontier* is a maximal subset of the Pareto-optimal frontier such that no two solutions in it have the same cost. Given approximation factor ε, an *ε-approximate Pareto-optimal solution set* Π_ε is a subset of the Pareto-optimal frontier such that , for any Pareto-optimal solution π for P, there exists a solution $\pi' \in \Pi_\varepsilon$ that $\pi' \preceq_\epsilon \pi$. Note that, given $\varepsilon = 0$, any cost-unique Pareto-optimal solution set is qualified as an ε-approximate Pareto-optimal solution set.

We define the domination factor of a path π' over another path π as

$$\text{DF}(\pi', \pi) = \max\left\{\frac{f_1(\pi')}{f_1(\pi)} - 1, \frac{f_2(\pi')}{f_2(\pi)} - 1, 0\right\},$$

which measures how "good" π' approximates π. It is easy to verify that $\pi' \preceq_\epsilon \pi$ for and only for $\varepsilon \geq \text{DF}(\pi', \pi)$. For a set of solutions, denoted as Π, we define its *approximation factor* as

$$\varepsilon(\Pi) = \max_{\pi \in \Pi^*}\left\{\min_{\pi' \in \Pi} \text{DF}(\pi', \pi)\right\}. \tag{1}$$

We slightly abuse the ε notation and use it as a function here. Roughly speaking, for each Pareto-optimal solution π, we find a path π' in Π that approximately dominates π the best, compute the value of the domination factor, and then take the maximum of these values over all Pareto-optimal solutions. Π is an ε-approximate Pareto-optimal solution set for and only for $\varepsilon \geq \varepsilon(\Pi)$.

[1] Therefore, ε is controlled by the algorithm and not user-specified.

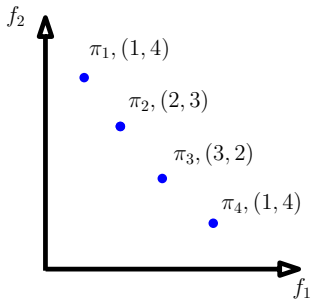


Figure 1: The $\mathbf{f}$-values of a Pareto-optimal frontier which consists of four paths π_1, π_2, π_3 and π_4. The numbers inside each bracket are the f_1 and f_2 values of each path.

Example 1. *Fig. 1 shows a Pareto-optimal frontier with four paths, denoted as* Π^*. *Let* $\Pi = \{\pi_1, \pi_2, \pi_4\}$ *denote a subset of* Π^*. *For any path* $\pi \in \Pi$, $\min_{\pi' \in \Pi} \mathrm{DF}(\pi', \pi) = 0$ *because* $\mathrm{DF}(\pi', \pi) = 0$ *by definition when* $\pi' = \pi$. *For path* π_3, $\mathrm{DF}(\pi', \pi_3)$, $\pi' \in \Pi$, *is minimized when* $\pi' = \pi_2$, *and we have* $\mathrm{DF}(\pi_2, \pi_3) = 0.5$. *Therefore,* Π *is a* 0.5*-approximate Pareto-optimal frontier.*

3 BOA* and BOA*$_\varepsilon$

BOA* (Hernandez et al. 2020) is a best-first bi-objective search algorithm. It maintains an OPEN list, containing the frontier of the search tree (i.e., the generated but not-yet-expanded nodes) and a set of solutions $sols$ that it has found so far. In BOA*, a node represents a path π from s_{start} to some state. In this paper, we use node and path interchangeably. For each state s, BOA* uses $g_2^{\min}(s)$ to store the minimum value of g_2 of all expanded paths that end at state s. Alg. 1 shows the pseudo-code of BOA* if all the colored text (which we will explain shortly) is ignored. At the beginning, OPEN contains only one path $[s_{\text{start}}]$. At each iteration, BOA* extracts a path with the lexicographically smallest $\mathbf{f}$-value among all paths in OPEN. A path π is pruned if there exists (i) an expanded path π' with the same ending state as π, i.e., $s(\pi) = s(\pi')$, and $\mathbf{g}(\pi') \preceq \mathbf{g}(\pi)$ or (ii) an expanded path π' with $s(\pi') = s_{\text{goal}}$ and $\mathbf{f}(\pi') \preceq \mathbf{f}(\pi)$. Hernandez et al. (2020) showed that testing (i) and (ii) can be done by testing if

$$g_2(\pi) \leq g_2^{\min}(s(\pi))$$

and if

$$f_2(\pi) \leq g_2^{\min}(s_{\text{goal}}), \tag{2}$$

respectively. If $s(\pi) = s_{\text{goal}}$, BOA* adds π to $sols$. Otherwise, BOA* expands π and generates a child node for each successor s' of $s(\pi)$. The child node represents the path that extends π with edge $\langle s, s' \rangle$.

Goldin and Salzman (2021) showed that by a slight modification, BOA* can be made to find an approximate Pareto-optimal frontier. To do so, we simply replace Eq. 2 with

$$f_2(\pi) \leq (1+\varepsilon) g_2^{\min}(s_{\text{goal}}),$$

which means path π is pruned if its $\mathbf{f}$-value is approximately dominated by the cost of a found solution path. The resulting algorithm is called BOA*$_\varepsilon$. In Alg. 1, we highlight in blue the necessary changes for converting BOA* to BOA*$_\varepsilon$.

Algorithm 1: BOA*$_\varepsilon$

Input : A search problem $(S, E, \mathbf{c}, s_{\text{start}}, s_{\text{goal}})$
Output: A cost-unique Pareto-optimal frontier.

```
1  sols ← ∅
2  OPEN ← {[s_start]}
3  for each s ∈ S do
4  |  g_2^min(s) ← +∞
5  while OPEN ≠ ∅ do
6  |  π ← OPEN.extract_min()
7  |  if g_2(π) ≥ g_2^min(s(π)) ∨ (1+ε)f_2(π) ≥ g_2^min(s_goal)
   |    then
8  |  |  continue
9  |  g_2^min(s(π)) ← g_2(π)
10 |  if s(π) = s_goal then
11 |  |  sols .add(π)
12 |  |  continue
13 |  for each s' ∈ succ(s(π)) do
14 |  |  π' ← extend(π, ⟨s(π), s'⟩)
15 |  |  if g_2(π') ≥ g_2^min(s') ∨ (1+ε)f_2(π') ≥ g_2^min(s_goal)
   |  |    then
16 |  |  |  continue
17 |  |  Add π' to OPEN
18 return sols
```

4 A-BOA*$_\varepsilon$

Conceptually, BOA* can be viewed as an anytime algorithm. It can be stopped anytime and then return $sols$ as the set of solutions that it has found by far. However, BOA* finds Pareto-optimal solutions in lexicographic order, and the solutions it finds first are the ones with small c_1 and none of them has relatively smaller c_2. Therefore, if stopped early, the returned solution set of BOA* might not be a good approximation of the Pareto-optimal frontier in terms of its approximation factor (Eq. 1). On the other hand, we can repeat BOA*$_\varepsilon$ multiple times with a sequence of decreasing ε until $\varepsilon = 0$, and this process of repeatedly running BOA*$_\varepsilon$ can be viewed as a basic anytime bi-objective search algorithm. However, rerunning BOA*$_\varepsilon$ from scratch is not efficient since it repeats the work of the previous iteration.

This motivates us to propose our new anytime bi-objective search algorithm, called Anytime BOA*$_\varepsilon$ (A-BOA*$_\varepsilon$). As we will see shortly, our algorithm A-BOA*$_\varepsilon$ makes use of *intervals* to bookkeep its progress in the previous iteration and avoid repeating work. We start with the definition and properties of interval before we describe A-BOA*$_\varepsilon$.

4.1 Preliminaries—Intervals

Definition 1 (Interval). *Let* π_{tl} *and* π_{br} *be two Pareto-optimal solutions such that* $f_1(\pi_{tl}) < f_1(\pi_{br})$ *(and hence* $f_2(\pi_{tl}) > f_2(\pi_{br})$*). Furthermore, let* Π *be a set of paths from* s_{start} *(not necessarily ending at* s_{goal}*) such that,* $\forall \pi \in \Pi$, $f_1(\pi_{tl}) \leq f_1(\pi) < f_1(\pi_{br})$ *and* $f_2(\pi_{br}) \leq f_2(\pi) < f_2(\pi_{tl})$. *We define their corresponding interval as the 3-tuple* $I = \langle \pi_{tl}, \pi_{br}, \Pi \rangle$ *and refer to* π_{tl}, π_{br}, *and* Π *as the top-left path, the bottom right path, and the to-expand paths of* I, *respectively.*

The notion of an interval is visualized in Fig. 2. Roughly

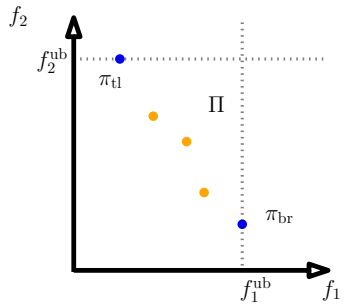


Figure 2: An example interval $I = \langle \pi_{\text{tl}}, \pi_{\text{br}}, \Pi \rangle$ in A-BOA$^*_\varepsilon$. The two blue dots represent the $\mathbf{f}$-values of paths π_{tl} and π_{br}, respectively. The orange dots represent the $\mathbf{f}$-values of all paths in Π.

Algorithm 2: Anytime BOA$^*_\varepsilon$

Input : A search problem $(S, E, \mathbf{c}, s_{\text{start}}, s_{\text{goal}})$, a consistent heuristic function, and parameter d for decreasing approximation factor
Output: A bounded approximate Pareto-optimal solution set

1 $\pi_{\text{tl}} \leftarrow$ a solution with the lexicographically smallest (c_1, c_2)
2 $\pi_{\text{br}} \leftarrow$ a solution with the lexicographically smallest (c_2, c_1)
3 $I_s \leftarrow \langle \pi_{\text{tl}}, \pi_{\text{br}}, \{[s_{\text{start}}]\} \rangle$
4 ILIST $\leftarrow [I_s]$
5 **while** $\exists I \in$ ILIST*:* $\hat{\varepsilon}(I) \neq 0$ **do**
6 $\quad I \leftarrow \text{argmax}_{I \in \text{ILIST}} \hat{\varepsilon}(I)$
7 $\quad$ ILIST$' \leftarrow$ Search$(I, \hat{\varepsilon}(I)/d)$
8 $\quad$ remove I from ILIST and add all intervals of ILIST$'$ to ILIST
9 **return** *ExtractSols(I)*

speaking, Π is a set of paths whose $\mathbf{f}$-value fall "between" the costs of π_{tl} and π_{br}.

Definition 2 (Approximation Factor of Interval). *Given an interval $I = \langle \pi_{tl}, \pi_{br}, \Pi \rangle$, we define its approximation factor $\hat{\varepsilon}(I)$ as*

$$\hat{\varepsilon}(I) = \max_{\pi \in \Pi} \{\min(DF(\pi_{tl}, \pi), DF(\pi_{br}, \pi))\},$$

if $\Pi \neq \emptyset$, otherwise, $\hat{\varepsilon}(I) = 0$.

Note that, given an interval $I = \langle \pi_{\text{tl}}, \pi_{\text{br}}, \Pi \rangle$, it holds by definition that, for any path $\pi \in \Pi$, π is $\hat{\varepsilon}(I)$-dominated by either π_{tl} or π_{br}.

As we will see shortly, A-BOA$^*_\varepsilon$ maintains a list of *intervals* ILIST, and, for each interval $\langle \pi_{\text{tl}}, \pi_{\text{br}}, \Pi \rangle$ in ILIST, Π contain all generated but not-yet-expanded paths that have the potential of being extended to a Pareto-optimal solution π_{sol} such that $f_1(\pi_{\text{tl}}) \leq f_1(\pi_{\text{sol}}) < f_1(\pi_{\text{br}})$. Although A-BOA$^*_\varepsilon$ does not explicitly maintain OPEN, the to-expand paths of intervals in ILIST can be conceptually viewed as "OPEN nodes" grouped by their $\mathbf{f}$-values.

Algorithm 3: Search

Input : An interval $I = \langle (\pi_{\text{tl}}, \pi_{\text{br}}, \Pi) \rangle$ and approximate factor ε
Output: A list of intervals, each with an approximation factor less than ε.

1 OPEN $\leftarrow \Pi$, $\Pi_{\text{tmp}} \leftarrow \emptyset$, $\pi_{\text{tmp}} \leftarrow \pi_{\text{tl}}$
2 **for each** $s \in S$ **do**
3 $\quad g_2^{\min}(s) \leftarrow \infty$
4 **while** OPEN $\neq \emptyset$ **do**
5 $\quad \pi \leftarrow$ OPEN.extract_min()
6 $\quad$ **if** `is_dominated`(π) **then**
7 $\quad\quad$ **continue**
8 $\quad g_2^{\min}(s(\pi)) \leftarrow g_2(\pi)$
9 $\quad$ **if** $s(\pi) = s_{goal}$ **then**
10 $\quad\quad$ add $\langle \pi_{\text{tmp}}, \pi, \Pi_{\text{tmp}} \rangle$ to ILIST
11 $\quad\quad \pi_{\text{tmp}} \leftarrow \pi$, $\Pi_{\text{tmp}} \leftarrow \emptyset$
12 $\quad\quad$ **continue**
13 $\quad$ **for each** $t \in succ(s(\pi))$ **do**
14 $\quad\quad \pi' \leftarrow \pi$ extended by s'
15 $\quad\quad$ **if** `is_dominated`(π') **then**
16 $\quad\quad\quad$ **continue**
17 $\quad\quad$ add π' to OPEN
18 Add $\langle \pi_{\text{tmp}}, \pi_{\text{br}}, \Pi_{\text{tmp}} \rangle$ to ILIST
19 **return** ILIST

20 **Define** `is_dominated`(π)**:**
21 $\quad$ **if** $f_1(\pi) \geq f_1(\pi_{br}) \vee f_2(\pi) \geq f_2(\pi_{tl}) \vee$ `can_prune_wsh`(π) **then**
22 $\quad\quad$ **return** True
23 $\quad$ **if** $g_2(\pi) \geq g_2^{\min}(s(\pi)) \vee (1+\varepsilon) f_2(\pi) \geq g_2^{\min}(s_{goal})$ **then**
24 $\quad\quad$ **if** $g_2(\pi) < g_2^{\min}(s(\pi)) \wedge f_2(\pi) < g_2^{\min}(s_{goal})$ **then**
25 $\quad\quad\quad$ add π to Π_{tmp}
26 $\quad\quad$ **return** True
27 $\quad$ **return** False

4.2 Algorithmic framework

Alg. 2 shows the pseudo code of A-BOA$^*_\varepsilon$. The algorithm initializes ILIST with only one interval $I_s = \langle \pi_{\text{tl}}, \pi_{\text{br}}, \{[s_{\text{start}}]\} \rangle$ with π_{tl} and π_{br} being solutions that have the lexicographically smallest (c_1, c_2) and (c_2, c_1), respectively (Line 4). Such solutions can be found using any single-objective graph search algorithm such as A* (Hart, Nilsson, and Raphael 1968). A-BOA$^*_\varepsilon$ then repeatedly picks the interval in ILIST which has the largest approximation factor and invoke Alg. 3 to continue search on its to-expand paths.

Alg. 3 takes interval I and an approximation factor ε as inputs and returns a set of new intervals, each with an approximation factor not larger than ε. In A-BOA$^*_\varepsilon$, the input ε for Alg. 3 is set to $\hat{\varepsilon}(I)/d$, where d is a parameter that specifies how fast the input approximation factor decreases in A-BOA$^*_\varepsilon$. Note that different strategies of decreasing ε could be easily implemented for A-BOA$^*_\varepsilon$ but are out of the scope of this work. A-BOA$^*_\varepsilon$ terminates when all intervals in ILIST have an approximation factor of zero. As we will show in Sec. 6, this means that the algorithm has found a cost-unique Pareto-optimal frontier. At any time point, the solutions that A-BOA$^*_\varepsilon$ has found so far can be reconstructed

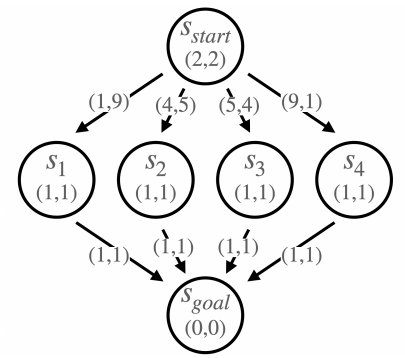


Figure 3: An example problem instance whose Pareto-optimal frontier consists of four paths. The pair of numbers inside each state is its $\mathbf{h}$-value. The pair of numbers annotate each edge is its costs.

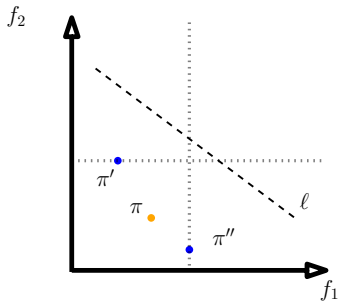


Figure 4: An example of pruning using the weight-sum heuristic. π is the path that we consider for pruning. π' and π'' are two known solutions. In this example, a weighted-sum heuristic function indicates that any potential solution from π must be on the upper-right side of line ℓ.

from ILIST as the set of top-left and bottom-right paths of all intervals. This set of solutions is a $\max\{\hat{\varepsilon}(I)|I \in \text{ILIST}\}$-approximate Pareto-optimal frontier.

Alg. 3 is similar to BOA$^*_\varepsilon$ with only differences in initialization and dominance checking. Alg. 3 initializes OPEN with to-expand paths of the input interval so that it does not start from scratch. Dominance checking of Alg. 3 is encapsulated in function `is_dominated` (Lines 20-27). Aside from applying the pruning conditions of BOA$^*_\varepsilon$ on Lines 23-26, Alg. 3 also prunes a path if it is dominated by either the top-left path or the bottom-right path of the input interval (Lines 21-22). Alg. 3 also stores those paths whose $\mathbf{f}$-values are ε-dominated by a solution that has been found but do not satisfy the pruning conditions of BOA* (Lines 24-25). Such paths are safe to prune for the input ε but still have the potential of being extended to Pareto-optimal solutions. A-BOA$^*_\varepsilon$ temporally stores these paths in Π_{tmp}. Each time a solution is found, a new interval is created with the previously found solution π_{tmp} as its top-left path, the newly found solution as its bottom-right path, and Π_{tmp} as its to-expand paths, π_{tmp} is set to π_{br} and Π_{tmp} is emptied. When OPEN is empty, Alg. 3 creates a new interval with the previously found solution π_{tmp} as its top-left path, π_{br} as its bottom-right path, and Π_{tmp} as its to-expand paths and returns the list of all new intervals. Each of these intervals has an approximation factor not larger than the input ε because all of its to-expand paths are ε-dominated by its top-left path.

Example 2. *Fig. 3 shows a problem instance whose Pareto-optimal frontier consists of four paths. Assume the parameter d for A-BOA$^*_\varepsilon$ is set to 4. At the beginning, A-BOA$^*_\varepsilon$ finds two Pareto-optimal paths $\pi_1 = [s_{start}, s_1, s_{goal}]$ and $\pi_4 = [s_{start}, s_4, s_{goal}]$ and initializes* ILIST *with interval $I_s = \langle \pi_1, \pi_4, \{[s_{start}]\}\rangle$. The $\mathbf{f}$-values for π_1, π_4, and path $[s_{start}]$ are $(2, 10)$, $(10, 2)$, and $(2, 2)$, respectively. The approximation factor for I_s is $\hat{\varepsilon}(I_s) = \min(DF(\pi_1, [s_{start}]), DF(\pi_4, [s_{start}])) = 4$.*

In the first iteration, A-BOA$^*_\varepsilon$ invokes Alg. 3 with I_s and $\varepsilon = 1$. Alg. 3 expands path $[s_{start}]$ and generates paths $[s_{start}, s_1]$, $[s_{start}, s_2]$, $[s_{start}, s_3]$, and $[s_{start}, s_4]$. Paths $[s_{start}, s_1]$ and $[s_{start}, s_4]$ are pruned on Lines 21-22 because they are weakly dominated by π_1 and π_4, respectively. A-BOA$^*_\varepsilon$ then expands $[s_{start}, s_2]$ and $\pi_2 = [s_{start}, s_2, s_{goal}]$ in sequence and finds a new solution π_2. $g_2^{min}(s_{goal})$ is updated to $g_2(\pi_2) = 6$. Path $[s_{start}, s_3]$ is pruned and put into Π_{tmp}. Alg. 3 returns two intervals $I_1 = \langle \pi_1, \pi_2, \emptyset\rangle$ and $I_2 = \langle \pi_2, \pi_4, \{[s_{start}, s_3]\}\rangle$. Thus, up until this point,, A-BOA$^*_\varepsilon$ has found three Pareto-optimal paths π_1, π_2, and π_4.

We have $\hat{\epsilon}(I_1) = 0$ by definition and $\hat{\epsilon}(I_2) = \min(DF(\pi_2, [s_{start}, s_3]), DF(\pi_4, [s_{start}, s_3])) = 0.2$. In the second iteration, A-BOA$^*_\varepsilon$ invokes Alg. 3 with I_2 and $\varepsilon = 0.05$. Eventually, Alg. 3 finds solution $\pi_3 = [s_{start}, s_3, s_{goal}]$ and returns two intervals $\langle \pi_2, \pi_3, \emptyset\rangle$ and $\langle \pi_3, \pi_4, \emptyset\rangle$. A-BOA$^*_\varepsilon$ terminates with all four Pareto-optimal paths found.

5 Weighted-Sum Heuristic

In this section, we describe a novel pruning technique for A-BOA$^*_\varepsilon$ that allows to discard nodes that cannot be extended to a Pareto-optimal solution. Many bi-objective search algorithms (Hernandez et al. 2020; Goldin and Salzman 2021) use a heuristic function to estimate the cost-to-go from a given state to the goal state for each of the two objectives. Computing such heuristics corresponds to a single-objective search from the goal state, and this process is very cheap (from a computational point of view) compared to solving the bi-objective search problem. This motivates us to design other heuristic functions that can be computed via single-objective search to further aid bi-objective search. Note that the techniques we are about to present does not use heuristic functions to guide the search but to prune away nodes.

Let $w > 0$ be a a real-valued parameter. *Weighted-sum heuristic* function $h_w : S \to \mathbb{R}$ estimates the weighted sum of costs $c_1 + w \cdot c_2$ from a given state to the goal state. By running a single-objective search algorithm such as Dijkstra's from the goal state using $c_1 + w \cdot c_2$ as the cost of each edge, the exact minimum weighted-sum value can be obtained and used as an admissible heuristic. In the rest of the paper we limit the discussion to admissible weighted-sum heuristics.

Property 1. *Let π be some path considered by a bi-objective search algorithm and let π' and π'' be two Pareto-optimal solutions that satisfy (i) $f_1(\pi') \leq f_1(\pi)$ and (ii) $f_2(\pi'') \leq$*

$f_2(\pi)$. *If*

$$g_1(\pi) + w * g_2(\pi) + h_w(s(\pi)) \geq f_1(\pi'') + w * f_2(\pi'),$$

then no path extending π can be included in the solution set.

Proof. Let π_{sol} be a solution path that extends π. Because the heuristic function $\mathbf{h}$ is consistent, we have $f_1(\pi_{\text{sol}}) \geq f_1(\pi) \geq f_1(\pi')$ and $f_2(\pi_{\text{sol}}) \geq f_2(\pi) \geq f_2(\pi'')$. Assuming that π_{sol} is eventually included in the solution set, π_{sol} is not weakly dominated by either π' or π''. Therefore, we have

$$f_2(\pi_{\text{sol}}) < f_2(\pi') \text{ and } f_1(\pi_{\text{sol}}) < f_1(\pi'). \tag{3}$$

Since h_w is admissible, we have

$$\begin{aligned} f_1(\pi_{\text{sol}}) + w * f_2(\pi_{\text{sol}}) &\geq g_1(\pi) + w * g_2(\pi) + h_w(s(\pi)) \\ &\geq f_1(\pi'') + w * f_2(\pi'), \end{aligned}$$

which contradicts with Eqs. 3.

□

Fig. 4 shows a visualization of how Property 1 can be used for pruning. Here, π, π', and π'' are paths as defined in Property 1. With only the information of h_1 and h_2, π still seems to have the potential to be extended to an undominated solution and hence cannot be pruned. Now, assume that line ℓ, depicted in Fig. 4 represents the equation

$$f_1 + w * f_2 = g_1(\pi) + w * g_2(\pi) + h_w(s(\pi)).$$

Since h_w is admissible, we know that the $\mathbf{f}$-value of any solution that extends π d must lie in the upper-right half-space of ℓ. From Fig. 4, we can see that point $(f_1(\pi''), f_2(\pi'))$ lies in the bottom-left half-space, which means that no such solution can be dominated by neither π' nor π'', and hence π is safe to prune. Therefore, with a weighted-sum heuristic function, the search algorithm is able to prune more paths.

For Alg. 3 to incorporate this pruning technique, it can use π_{tmp} and π_{br} as π' and π'', respectively. A path π can be pruned if, given weighted-sum heuristic h_w,

$$g_1(\pi) + w * g_2(\pi) + h_w(s(\pi)) \geq f_1(\pi_{\text{br}}) + w * f_2(\pi_{\text{tmp}}).$$

In Alg. 3, we highlight in blue where the pruning technique can be applied. Since BOA* expands nodes with a lexicographic order, it is not straightforward to see which path can be used as π'' for the new pruning technique. We leave it as future work of how to incorporate BOA* with weighted-sum heuristics.

6 Theoretical Results

In this section, we provide some theoretical results of A-BOA$^*_\varepsilon$. Specifically, after describing some properties of ILIST and intervals, we show that (i) the set of solutions that A-BOA$^*_\varepsilon$ finds at any point of time is an $\max\{\hat{\varepsilon}(I)|I \in \text{ILIST}\}$-approximate Pareto-optimal frontier (Thm. 1) and that (ii) A-BOA$^*_\varepsilon$ eventually finds a cost-unique Pareto-optimal frontier (Thm. 2).

Property 2. *Let $[I_1, I_2 \dots I_M]$ denote* ILIST sorted in ascending order according to the f_1 value of the top-left path of each interval. For any $i \in \{1, 2 \dots M-1\}$, the bottom-right path of I_i is also the top-left path of I_{i+1}.

Proof sketch. The property is trivially true at the beginning when $M = 1$. In each iteration, an interval I is chosen, removed from ILIST and sent to Alg. 3. In Alg. 3, intervals are created with increasing f_1 values of the top-left paths. when an interval is created, its bottom-right path is stored in π_{tmp} and becomes the top-left path of the next created interval. Moreover, the first and the last created intervals have the same top-left path and the same bottom-right path as I, respectively. Therefore, after replacing I with the list of intervals return by Alg. 3 in ILIST, Property 2 still holds. □

Conceptually, the to-expand paths of each interval can be viewed as the paths that has the potential of being extended to Pareto-optimal paths with certain cost range, and this cost range is specified by the top-left and the bottom-right paths of the interval. Property 2 further shows that these cost ranges of intervals in ILIST are disjoint. Therefore, A-BOA$^*_\varepsilon$ can search each interval independently using Alg. 3.

Lemma 1. *When A-BOA$^*_\varepsilon$ prunes a path in Alg. 3 and this prevents it in the future from adding a solution π_{sol} to the solution set, then it can still add in the future a solution that weakly dominates this path.*

Proof sketch. There are three cases when A-BOA$^*_\varepsilon$ prunes a path π:

1. It prunes π on Line 21 of Alg. 3. Then, A-BOA$^*_\varepsilon$ has already found a solution, that is, either π_{tl} or π_{br}, that weakly dominates π (and hence π_{sol}). This solution is already included in the solution set.
2. It prunes π on Line 26 of Alg. 3 after storing this path in π_{tmp}. Then, the pruned path will be stored as a to-expand path for an interval and extracted again in the future for expansion.
3. It prunes π on Line 26 of Alg. 3 without storing this path in π_{tmp}. Then, path π satisfies that $g_2(\pi) \geq g_2^{min}(s(\pi)) \vee f_2(\pi) \geq g_2^{min}(s_{\text{goal}})$. Note that this is the same pruning condition as the one in BOA*, and the proof of Lemma 7 of Hernandez et al. (2020) can be applied here.

□

Lemma 2. *In A-BOA$^*_\varepsilon$, the top-left and the bottom-right paths of any interval are always Pareto-optimal solutions.*

Proof sketch. Roughly from Property 2 and the fact that Alg. 3 expands node in a lexicographic order and also uses the pruning conditions of BOA*, we can prove that the solution paths that A-BOA$^*_\varepsilon$ * finds do not weakly dominate each other. Assume a solution, denoted by π_{sol}, is found by Alg. 3 and is dominated by a solution, denoted by π'. From Lemma 1, A-BOA$^*_\varepsilon$ can eventually adds a solution that weakly dominates π' (and hence π_{sol}), which cause a contradiction.

□

The following two theorems show that A-BOA$^*_\varepsilon$ finds an approximate Pareto-optimal frontier with a theoretical approximation factor guarantee and eventually finds a cost-unique Pareto-optimal frontier.

Theorem 1. *The set of solutions that A-BOA$^*_\varepsilon$ finds at any point of time is a* $\max\{\hat{\varepsilon}(I)|I \in \text{ILIST}\}$*-approximate Pareto-optimal frontier.*

Proof sketch. Let Π denote the set of solutions that A-BOA$^*_\varepsilon$ has found and Π^* denote the Pareto-optimal frontier. From Lemma 1, for any $\pi \in \Pi^* \setminus \Pi$, there exists a to-expand path, denoted as $\hat{\pi}$, that can be extended to a solution that weakly dominates π. Because A-BOA$^*_\varepsilon$ uses consistent heuristic functions, we have $\mathbf{f}(\hat{\pi}) \preceq \mathbf{f}(\pi)$, and hence $DF(\pi', \pi) \leq DF(\pi', \hat{\pi})$ for any path π'. Let $I = \langle \pi_{\text{tl}}, \pi_{\text{br}}, \Pi \rangle$ denote the interval $\hat{\pi}$ is from. We can prove that $\min_{\pi' \in \Pi} \{DF(\pi', \hat{\pi})\} = \min(DF(\pi_{\text{tl}}, \hat{\pi}), DF(\pi_{\text{br}}, \hat{\pi}))$. Intuitively, from Property 1, we can see that, among Π, either π_{tl} or π_{br} is the path whose cost approximates $\mathbf{f}(\hat{\pi})$ the best. We can further prove that $\min_{\pi' \in \Pi} \{DF(\pi', \hat{\pi})\} \leq \hat{\varepsilon}(I)$ using the definition of $\hat{\varepsilon}(I)$. To summarize, $\min_{\pi' \in \Pi} \text{DF}(\pi', \pi)$ is upper-bounded by the approximation factor of some interval in ILIST. Therefore, $\epsilon(\Pi)$ is upper-bounded by $\max\{\hat{\varepsilon}(I)|I \in \text{ILIST}\}$. □

Theorem 2. *A-BOA$^*_\varepsilon$ eventually finds a cost-unique Pareto-optimal frontier.*

Proof sketch. Since the graph is finite, the number of distinctive costs of all paths from s_{start} whose $\mathbf{f}$-value is not dominated by any Pareto-optimal solution is finite. When ε is sufficiently small, none of the found solutions can ε-dominates any such path. No path is stored as a to-expand paths, and hence all intervals in ILIST have an approximation factor of zero and Alg. 2 terminates. □

7 Experiment Results

In this section, we evaluate A-BOA$^*_\varepsilon$ empirically by comparing it with the following algorithms:

1. BOA*.
2. Basic-A-BOA$^*_\varepsilon$: Basic-A-BOA$^*_\varepsilon$ iteratively invokes BOA$^*_\varepsilon$, each time with the input approximation factor divided by a constant number d. In each iteration, Basic-A-BOA$^*_\varepsilon$ identifies if there is a pruned path that has the potential to be extended to a Pareto-optimal solution and terminates if no such path exists. Basic-A-BOA$^*_\varepsilon$ is similar to A-BOA$^*_\varepsilon$ except that it does not reuse previous work.
3. Iterative-PP-A*: Iterative-PP-A* is based on PP-A*, a state-of-the art approximate bi-objective search algorithm. Given problem instance P and approximation factor ε, PP-A* finds a set of solutions Π such that, for any solution π of P, there exists a solution $\pi' \in \Pi$ that ε-dominates π. Note that π' is not necessarily a Pareto-optimal solution. [2] Arguably, PP-A* uses more complicated data structures when compared to BOA* and hence it is more difficult to reuse search effort of PP-A*. In our experiments, Iterative-PP-A* iteratively invokes PP-A* with a decreasing sequence of $\varepsilon = [0.01, 0.001, 0.0001, 0]$.

We run two variants of A-BOA$^*_\varepsilon$, one with the weighted-sum heuristic pruning technique (denoted as A-BOA$^*_\varepsilon$-w) and one without this optimization (denoted as A-BOA$^*_\varepsilon$). All algorithms were implemented in C++ and shared the code base as much as possible.[3]

We use three road maps of the 9th DIMACS Implementation Challenge[4], which are BAY, FLA, and NE. For each map, we generated 25 instances with randomly selected start and goal states. The heuristic values are the exact minimum costs to the goal state for each single objective, computed with Dijkstra's algorithm. The reported runtime does not include this computation. However, the reported runtime does include the computation of weighted-sum heuristic if an algorithm makes use of it. For all algorithms, we set a five-minute runtime limit for solving each problem instance.Parameter d for both Basic-A-BOA$^*_\varepsilon$ and A-BOA$^*_\varepsilon$ is set to 4. A-BOA$^*_\varepsilon$-w uses the weighted-sum heuristic function with $w = 1$. We ran all experiments on a MacBook Pro with an M1 Pro chip and 32GB of memory.

Table 1 shows the number of instances that are solved (that is, the algorithm finds a cost-unique Pareto-optimal frontier for the instance) within the time limit, the average runtime (in seconds, where timeout instances are counted as five minutes), and the number of expanded nodes for different algorithms. While using similar schemes for decreasing the approximation factor, Basic-A-BOA$^*_\varepsilon$ has much larger average runtimes and numbers of node expansions than A-BOA$^*_\varepsilon$ and A-BOA$^*_\varepsilon$-w in all instance, which is caused by the redundant work it does. Iterative-PP-A* has better runtime results than the one of Basic-A-BOA$^*_\varepsilon$ because it runs PP-A* for much fewer (=4) iterations. However, it still takes longer time for Iterative-PP-A* to converge to unique-cost Pareto-optimal frontiers than BOA* and variants of A-BOA$^*_\varepsilon$. A-BOA$^*_\varepsilon$-w outperforms A-BOA$^*_\varepsilon$ on all three maps. It also outperforms BOA* in terms of numbers of node expansions on all three maps and outperforms BOA* in terms of average runtime on FLA and NE. The results show that, for larger maps and more difficult instances, the computational overhead of computing weighted-sum heuristic is outweighed by its benefits on improving the total runtime.

Fig. 5 shows the behaviours of all algorithms in three problem instances, each from a different road network. These three instances can be viewed as representative cases for problem instances of different difficulties. In all three cases, A-BOA$^*_\varepsilon$, A-BOA$^*_\varepsilon$-w and Basic-A-BOA$^*_\varepsilon$ finds solution sets with better approximation factor than BOA* initially, and BOA* outperforms A-BOA$^*_\varepsilon$-w only in the easiest instance (instance (a)) In all three instances, Basic-A-BOA$^*_\varepsilon$ is slower than both variants of A-BOA$^*_\varepsilon$. A-BOA$^*_\varepsilon$-w has a similar behaviour as A-BOA$^*_\varepsilon$ at the beginning and is significantly faster than A-BOA$^*_\varepsilon$ when approximation factor becomes small, which shows that the weighted-sum heuristic

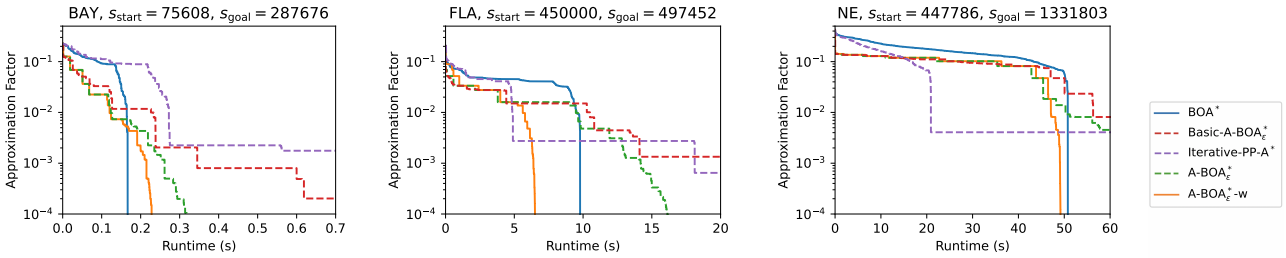
[2] This statement contradicts with the original statement for PP-A* (Goldin and Salzman 2021). We confirmed with the main author of the PP-A* paper, and he acknowledged that the original statement was inaccurate.

[3] Our code and data will be publicly available upon publication of the paper.

[4] http://users.diag.uniroma1.it/challenge9/download.shtml

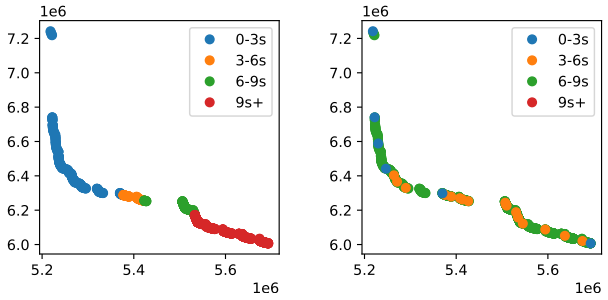
	BAY $\lvert sols \rvert = 102$ on average			FLA $\lvert sols \rvert = 457$ on average			NE $\lvert sols \rvert = 930$ on average		
	Solved	Runtime	#Exp	Solved	Runtime	#Exp	Solved	Runtime	#Exp
BOA*	25/25	**0.20**	161K	25/25	4.17	1,985K	25/25	37.28	10,072K
Basic-A-BOA$^*_\varepsilon$	25/25	9.45	1,641K	19/25	97.80	14,194K	15/25	178.57	29,567K
Iterative-PP-A*	25/25	2.41	530K	23/25	45.44	4,124K	18/25	123.23	9,517K
A-BOA$^*_\varepsilon$	25/25	0.34	271K	25/25	5.74	2,843K	23/25	61.23	12,403K
A-BOA$^*_\varepsilon$-w	25/25	0.29	**153K**	25/25	**3.80**	**1,759K**	25/25	**34.58**	**8,729K**

Table 1: Experimental results for different road maps.



(a) BAY, $s_{start} = 75608$, $s_{goal} = 287676$ (b) FLA, $s_{start} = 450000$, $s_{goal} = 497452$ (c) NE, $s_{start} = 447786$, $s_{goal} = 1331803$

Figure 5: Behaviors of different algorithms on three representative problem instances. The x-axis represents the elapsed time, and each line shows the approximation factor of the solution set an algorithm has found over time. The faster the approximation factor decreases the better.



(a) BOA* (b) A-BOA$^*_\varepsilon$-w

Figure 6: The solution sets that BOA* and A-BOA$^*_\varepsilon$-w find in the problem instance of Fig. 5b. Solutions are marked with different colors according to the elapsed time that they are found by the algorithm.

pruning is most effective in the later stage of problem solving. In both instances (b) and (c), Iterative-PP-A* is the first algorithm which finds a solution set with approximation factor less than 0.01. This is because PP-A* does not guarantee to find Pareto-optimal solutions, and hence solves an arguably easier problem than the other four algorithms. However, since Iterative-PP-A* does not reuse any previous search effort, it is slower than A-BOA$^*_\varepsilon$ and A-BOA$^*_\varepsilon$-w in terms of further decreasing the approximation factor to 0.001 or below.

Fig. 6 shows the Pareto-optimal frontier of the problem instance in Figure 5b, marked with different colors according to the elapsed time that each solution is found. BOA* finds solutions in a strictly lexicographic order while A-BOA$^*_\varepsilon$-w first finds a set of solutions with diverse costs and then found more solutions to add to the solution set.

8 Conclusion

In this paper, we propose an anytime approximate bi-objective search algorithm, called A-BOA$^*_\varepsilon$. It efficiently reuses search efforts from previous iterations and makes use of a novel pruning technique. Our experimental results show that A-BOA$^*_\varepsilon$ significantly outperforms an anytime approximate bi-objective search algorithm that does not reuse previous search efforts. In limited time, A-BOA$^*_\varepsilon$ often finds solutions that have much better approximation factors than the ones found by BOA*.

There are several interesting directions for future work. One interesting direction is to develop anytime bi-objective search algorithms which find solutions that are not necessarily Pareto-optimal. By leveraging this, PP-A* is able to quickly find a set of solutions with small approximation factor. However, it is unclear to us how to efficiently reuse the search effort of PP-A*. Another interesting direction is to generalize anytime approximate bi-objective search algorithms to the search problem with more than two objectives.

References

Bachmann, D.; Bökler, F.; Kopec, J.; Popp, K.; Schwarze, B.; and Weichert, F. 2018. Multi-Objective Optimisation Based Planning of Power-Line Grid Expansions. *ISPRS International Journal of Geo-Information*, 7(7): 258.

Breugem, T.; Dollevoet, T.; and van den Heuvel, W. 2017.

Analysis of FPTASes for the multi-objective shortest path problem. *Computers & Operations Research*, 78: 44–58.

Bronfman, A.; Marianov, V.; Paredes-Belmar, G.; and Lüer-Villagra, A. 2015. The maximin HAZMAT routing problem. *European Journal of Operational Research*, 241(1): 15–27.

Cohen, L.; Greco, M.; Ma, H.; Hernández, C.; Felner, A.; Kumar, T. K. S.; and Koenig, S. 2018. Anytime Focal Search with Applications. In *IJCAI*, 1434–1441.

Ehrgott, M. 2005. *Multicriteria Optimization (2. ed.)*. Springer.

Fu, M.; Kuntz, A.; Salzman, O.; and Alterovitz, R. 2019. Toward Asymptotically-Optimal Inspection Planning Via Efficient Near-Optimal Graph Search. In *RSS*.

Fu, M.; Salzman, O.; and Alterovitz, R. 2021. Computationally-Efficient Roadmap-based Inspection Planning via Incremental Lazy Search. In *ICRA*, 7449–7456.

Goldin, B.; and Salzman, O. 2021. Approximate Bi-Criteria Search by Efficient Representation of Subsets of the Pareto-Optimal Frontier. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 149–158.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Hernandez, C. U.; Yeohz, W.; Baier, J. A.; Zhang, H.; Suazoy, L.; and and, S. K. 2020. A Simple and Fast Bi-Objective Search Algorithm. In *ICAPS*, 143–151.

Likhachev, M.; Gordon, G. J.; and Thrun, S. 2003. ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *NIPS*, 767–774.

Mandow, L.; and De La Cruz, J. L. P. 2010. Multiobjective A* search with consistent heuristics. *Journal of the ACM (JACM)*, 57(5): 1–25.

Pulido, F.-J.; Mandow, L.; and Pérez-de-la Cruz, J.-L. 2015. Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research*, 64: 60–70.

Stern, R.; Felner, A.; van den Berg, J.; Puzis, R.; Shah, R.; and Goldberg, K. 2014. Potential-based bounded-cost search and Anytime Non-Parametric A^*. *Artificial intelligence*, 214: 1–25.

Tsaggouris, G.; and Zaroliagis, C. D. 2009. Multiobjective Optimization: Improved FPTAS for Shortest Paths and Non-Linear Objectives with Applications. *Theory Comput. Syst.*, 45(1): 162–186.

van den Berg, J.; Shah, R.; Huang, A.; and Goldberg, K. Y. 2011. Anytime Nonparametric A*. In *AAAI*.

Warburton, A. 1987. Approximation of Pareto optima in multiple-objective, shortest-path problems. *Operations research*, 35(1): 70–79.